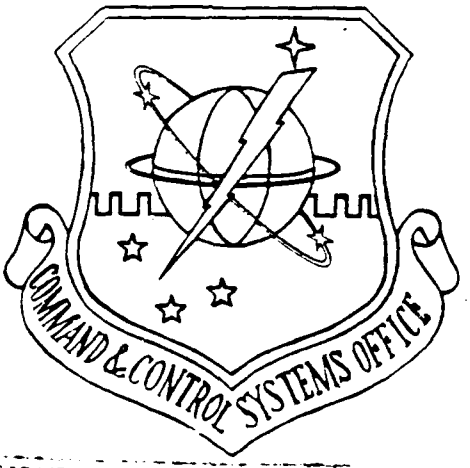
ADA* EVALUATION PROJECT

ADA* TRAINING FOR DEVELOPMENT TEAMS

Prepared for

HEADQUARTERS UNITED STATES AIR FORCE
Assistant Chief of Staff for Information System
Technology & Security Division



Prepared by
Standard Automated Remote to AUTODIN Host (SARAH) Branch
COMMAND AND CONTROL SYSTEMS OFFICE
Tinker Air Force Base
Oklahoma City, OK 73145

* Ada is a registered trademark of the U.S. Government
(Ada Joint Program Office)

13 March 1986

THIS REPORT IS THE FIRST OF A SERIES WHICH WILL DOCUMENT THE LESSONS LEARNED IN THE USE OF Ada IN A COMMUNICATIONS ENVIRONMENT.

ABSTRACT

This paper addresses training requirements for Ada development teams and reports on various training methods. The first section covers introductory material. Some background information is provided on the scope of the paper and the basis for evaluating training requirements. In addition, this section covers the purpose of the paper and outlines some assumptions and constraints. The second section describes Ada training needs. This section argues that to obtain the full benefits of Ada related technology, the training program should include training in software engineering, development methodologies, support environments and Ada language syntax. The necessity to train managers is also discussed. The third section describes training methods. The use of contractor, government and in-house training is covered. This section is based largely on the Standard Automated Remote to AUTODIN Host (SARAH) project training program. As such, it should provide practical information to organizations who are contemplating developing Ada based software systems. The last section summarizes the major points and makes recommendations on a possible approach to Ada training.

TABLE OF CONTENTS

ADA TRAINING FOR DEVELOPMENT TEAMS

1. INTRODUCTION

1.1. BACKGROUND

Software development using the Ada language and associated software engineering technology requires a higher degree of training than was required for older languages. There have been many articles outlining the benefits of using the Ada language [1,3,4]. Through Ada and associated software engineering technology we can gain significant cost benefits through code reuse, transportability, lower maintenance costs and increased productivity. Managers are quick to point out these benefits, yet many do not understand that without the correct engineering approach, extensive use of tools and a high level of managerial support, these gains will not be achieved. If development teams are to successfully develop Ada software in this very complex environment, they must receive training in a number of areas. In addition to training in Ada syntax, team members require training in software engineering, development methodologies, and programming support environments.

So that potential Ada developers could gain a practical insight into what was required to successfully develop Ada software, the Air Staff tasked the Command and Control Systems Office (CCSO) with evaluating the Ada language while developing real-time digital communications software. The evaluation was to consist of a number of evaluation papers, one of which was to deal with training requirements. CCSO chose the Standard Automated Remote to Automatic Digital Network (AUTODIN) Host (SARAH) project as the basis for this evaluation. SARAH is a small to medium size project (approx. 40,000 lines of source code) which will function as a standard intelligent terminal for AUTODIN users and will be used to help eliminate punched cards as a transmit/receive medium [8]. The development environment for SARAH consists of the SOFTECH Ada Language System (ALS) hosted on a Digital Equipment Corporation VAX 11/780, ALSYS Ada compiler for the IBM PC/AT, a Burroughs XE550 Megaframe and several IBM compatible PC-XT and PC-AT microcomputers. The ALS environment is the focal point of this integrated development environment. The source code developed on the XE550 and microcomputer workstations is maintained by the ALS configuration control system and will be transferred to the PC-ATs for final compilation and targeting. The SARAH software targets are the IBM compatible PC-AT and PC-XT microcomputers.

The SARAH team required training in several areas. These included training on the Ada language, on the latest design methodologies, on the ALS Ada Programming Support Environment (APSE), and staff management and analyst training. Since the chief designers were well versed in modern software engineering practices prior to project initiation, no software engineering

training was provided.

Several methods were used to provide the required training. Formal training was obtained from both commercial and government sources. In-house training was conducted using a Computer Aided Instruction (CAI) package, instructional video tapes and lectures. The SARAH team also gained a great deal of practical knowledge through their involvement in the local and national Ada communities.

1.2. PURPOSE

The aims of this paper are to:

- Outline training needs for software development teams.
- Provide practical feedback on Ada training to prospective Ada developers.
- Make recommendations on a possible approach to Ada training.

1.3. ASSUMPTIONS AND CONSTRAINTS

The assumptions and constraints are as follows:

- A major constraint is the size of the SARAH project. Since the SARAH project team is small (10 persons), some of the experiences reported in this paper may not be appropriate for training larger development teams.
- The evaluation is based on a training budget of $50,000. This paper assumes that sufficient funds will be made available for team training.
- The SARAH team members had a variety of previous experience. Some members had very little software experience, others were well versed in assembly language programming and some were experienced Pascal programmers.
- Since the SARAH project is at the analysis/design stage of development, the effectiveness of the training received cannot be fully evaluated at this time. At the completion of the SARAH project, a summary paper will reflect on how well the training met our needs.

2. ADA TRAINING NEEDS

2.1. SOFTWARE ENGINEERING TRAINING

Software engineering must be stressed if software development using the Ada language is to be successful. The Ada language was designed by software engineers who based their design on modern software engineering principles [1]. Ada supports many of the features of modern software engineering. For example, Ada provides facilities for structured programming, strong data typing, separate compilation, information hiding, data abstraction, and procedural abstraction. These facilities, when properly applied by designers and programmers, can reduce maintenance costs, promote transportability, and improve reliability and survivability. These benefits are realized only when those facilities are used in the manner intended by its designers, otherwise the major benefits of the language will be lost.

Since many of Ada's features relate directly to modern software engineering, the language is easier to learn if it is presented in a manner that facilitates the implementation of these engineering principles and goals. The size of the Ada language has often been criticized; some authors have indicated that it would be beyond most programmers to ever gain a working knowledge of the language. Many of these comments were made in a comparison of Ada to traditional languages such as FORTRAN and COBOL. The comments are well founded if the same approach to teaching these second generation languages is used to teach Ada. Unlike traditional high order languages, Ada provides the engineer with language capabilities built in to facilitate the solution of a complex array of problems not available in other languages. Educators need to introduce each of these capabilities and explain their purpose. For example, an apprentice builder must be told that a saw is used for cutting wood and then shown the correct way to use the saw. Similarly, an Ada software engineer must be introduced to the concept of the package and then shown how this can be used for data abstraction. Students should understand that the Ada language itself does not solve problems. They should learn the purpose of each Ada engineering tool and how each can be used to successfully develop software systems. If Ada is taught in this way, students will more easily remember the syntax of the language because of its relationship to software system engineering.

In addition to introducing students to the facilities of software engineering, software engineering training must cover many other aspects of software development. For example, instruction should be given on software maintenance, configuration management, documentation, testing, software reuse and the use of programming support tools. As discussed, through software engineering we can effectively teach Ada syntax; however, the success of a project is determined not by the code alone. The scope of software training should cover the full

software lifecycle. The student must be made aware of the impact that a bad design will have on software maintenance. Similarly, if the configuration of a software product is not carefully controlled, the student should realize that the software could be made useless. If students are taught fundamental software engineering principles and techniques, organizations can gain significant long and short term cost savings. These gains will be realized because productivity is enhanced by software reuse and reduced maintenance efforts, software systems will be more easily transportable to different hardware, and software engineering-Ada trained personnel will quickly become productive when transferred to new Ada projects due to standardization of the software development process.

2.2. DEVELOPMENT METHODOLOGY TRAINING

Development methodology training should cover the different analysis/design methodologies as well as the use of program design languages (PDLs) [9]. There are many methodologies available for software development. Those most applicable for the development of Ada software are covered in the Methodman I [10] document. Since no one methodology currently covers the full software lifecycle or all design paradigms, students should be introduced to a number of the methodologies so that they can select the features of each which would best suit their needs. For example, Object Oriented Design (OOD) [12] provides a powerful method for design but does not effectively address the analysis phase. Also, OOD is largely ineffective for the design of real-time process driven systems. In these cases a methodology which supports process abstraction is required and so the designer may chose a methodology such as Jackson System Development (JSD) [13] or the Process Abstraction Method for Embedded Large Applications (PAMELA) [14]. Some cases may call for a multi-paradigm design [15] which requires the inclusion of the concepts of two or more methodologies. Clearly, courses teaching a single methodology do not prepare students for the variety of design and development problems that exist.

Software engineers must be able to stay current with advances in software development technology. The next few years will bring many new methodologies, many of which will be superior to those used today. Students should be introduced to the different design methods such as procedural, type and process abstraction so that they have the background for understanding the scope of these new methodologies. In addition, new methodologies will be developed which will more effectively cover the testing, integration and maintenance phases of software development. The limitations of existing methodologies should be highlighted so that future software engineers can more readily identify methodologies which will support full lifecycle needs.

Educators should cover the arguments governing the use of PDLs [9] and provide an introduction to the types of PDL currently in use. Students should be able to determine whether a

PDL is required and if so what type of PDL would best suit their needs. The whole area of PDL is being hotly debated. There are some who believe that a PDL is not required for Ada development. They suggest that the language itself can be used as a PDL. Those arguing for PDLs are in disagreement over what form it should take. If an organization elects to use a PDL, or a contract specifies that a PDL will be used, the training requirement should be assessed and the training budget should be adjusted appropriately [7].

2.3. SUPPORT ENVIRONMENT TRAINING

Training in the use of Ada Programming Support Environment (APSE) tools is required if an organization is to achieve productivity benefits. Software engineers cannot be expected to fully utilize these environments unless they are trained in their use. For example, the Ada Language System from SOFTECH is an APSE consisting of more than 70 tools [16]. The tools provide functions such as configuration management, symbolic debugging, frequency and timing analysis and code formatting. The environment is very complex and users require a high level of training if they are to use the tools effectively.

Students should first be given an overview of the environment. The overview should provide an introduction to the command language, database organization and file administration. In addition, the overview should cover basic operations such as invoking tools, compiling programs, exporting, and linking. After the students are familiar with these basic operations they should be given a more advanced user course which covers the tools in detail. Educators should introduce each tool, demonstrate its use and show how the tool could be used to improve quality and productivity. There is little doubt that significant productivity gains can be made through the use of automated tools; however, managers must understand that training is required if these tools are to be used effectively.

APSE administrator training will be required for effective use of the overall APSE system. The APSE administrator position on a development team is enormously important. The administrator is responsible for controlling access, transmission/reception of reusable library modules, providing incremental updates and database administration/maintenance. If these tasks are not done properly, the whole development effort could be placed in jeopardy. APSE administrator training is therefore an important part of an overall training program for Ada development teams.

Organizations should monitor the progress of the efforts aimed at standardizing support environments so that in the event that standardization becomes a reality, the training investment will support a number of APSE implementations. In particular, the Common APSE Interface Set (CAIS) standard [17] has received a high degree of attention. The introduction of a validation facility for support environments may mean that the CAIS standard

will gain a high level of acceptance. Although vendors will be permitted to provide additional advanced tools, the minimum toolset will be the same on each machine and the Kernel Ada Programming Support Environment will allow for the migration of tools between different hardware environments. If this occurs, the training received on an APSE which complies with the CAIS standard will allow software personnel to operate and use the tools of an APSE which may be hosted on a different system.

2.4. LANGUAGE TRAINING

Language features should be taught in conjunction with software engineering principles and goals [9]. As discussed earlier, Ada is the product of software engineering and supports many modern software engineering features. If the Ada language is broken up into the logical partitions which correspond to engineering building blocks, students will find it easier to understand and use the entire language. Moreover, the language was designed to be used in conjunction with development methodologies and programming environments [2]. Students should be introduced to these areas prior to being submitted to language training so that they can understand how different language features apply to the overall development process. If the full benefits of Ada are to be achieved then language training cannot be conducted in isolation.

The curriculum for Ada language training can quite effectively be divided into two separate areas: basic and advanced topics. After completing a basic course, the student should be able to:

- Declare and use Ada objects and types,
- Understand and formulate Ada statements,
- Code and call Ada subprograms,
- Design and use Ada packages,
- Raise and handle exceptions,
- Perform input/output.

The basic course should give the student a working knowledge of sequential Ada.

An advanced Ada course should cover the concurrent aspects of Ada and low level features. The course should stress important design features such as the use of generics for software reuse. Upon completion, students should be able to design, code, and test Ada programs that use generics, low level features, and tasks. Moreover, the course should allow students to apply sound software engineering principles to produce well-designed Ada systems.

The length of time required for formal Ada language training is dependent on the students' previous experience, the amount of pre-course preparation, and the requirement for practical training. Several organizations (e.g. Softech Inc. and the US Air

Force Air Training Command (ATC)) suggest that at least six weeks are required if this type of training is to be successful. These courses include a high degree of practical training and assume no previous experience in structured programming.

There are other training organizations that believe that the Ada language can successfully be covered with two 40 hour courses; however, several considerations need to be made if this training is to be effective. First, students should have at least some knowledge of the Ada language prior to commencing formal training. This could be achieved through the use of a CAI package, video tapes and self-study. Second, since a large amount of material needs to be covered in a relatively short time, practical training should be limited and structured towards reinforcing the major concepts. Third, consolidation time of at least one week should follow each course so that students can have the opportunity to consolidate their knowledge on practical exercises.

Student assessment should be provided by the training organization. These assessments are beneficial for a number of reasons. First, students are generally more motivated towards retaining information if they know that they will be tested at the end of a training period. They tend to compete more with their fellow students and do not like to see unfavorable reports sent to supervisors. Second, through student assessments, managers are given an insight into the effectiveness of the training program by monitoring student progress. Managers should be provided with a copy of the assessment scheme and results so that they can use the information to more effectively manage their software projects.

In summary, formal Ada language training could be covered in as little as 80 hours; however, the training must be intense, practical training must be well organized, time must be allocated for consolidation, and the students must be very motivated.

2.5. MANAGEMENT TRAINING

Software development using the Ada language and associated software engineering technology will only be successful if full support is provided by management. To do this, managers must have a firm understanding of the technology being used and they must be introduced to some of the problems that may be encountered during development. Managers at all levels within an organization should receive Ada technology training. The initial investment for Ada development is high and there are a number of potential pitfalls. Managers must be educated in this new technology so that they will have the ability to support development teams when problems arise. As with any new technology, there are many problems yet to be overcome and the organization will benefit only if management works together with development teams to solve these problems.

Managers should be introduced to the Ada community and provided with information on where they can find additional

information and help. There are many sources of Ada related information available to managers. For example, the Ada Joint Program Office (AJPO) operates the Ada Information Clearinghouse which distributes Ada related information. Moreover, various organizations have been formed to act as forums for Ada discussion (e.g. SIGAda and AdaJUG). Management training should cover this type of information so that managers can keep current with new advances in Ada technology.

The major benefits of Ada based technology should be covered so that the manager understands what can and should be achievable. Managers should be shown how the Ada language and associated software technology can lower maintenance costs, improve software transportability, and improve reliability. Software reuse should be covered. Managers should be shown that by designing software with reusability in mind, significant cost savings can be made. In addition, managers should be aware that libraries of reusable software such as SIMTEL-20 [18] now exist. If managers are made aware of how Ada based technology can be used to develop quality and cost effective software products, they will be in a better position to help introduce this new technology into their organization.

Managers should be informed that with Ada there is a high initial investment and that some of the benefits will only be realized in the long term. The capital investment is significantly different from what was required for older languages. For example, to obtain many of the productivity benefits associated with Ada, automated tools are required. Software maintenance cost will only be reduced if a proper development methodology is applied and so this translates into higher training costs. In addition, the length and cost of language training will be higher than for older languages. Software personnel will be more highly trained and so key personnel will most probably be more expensive to hire. The manager must be shown that the cost savings through less retraining, higher productivity and less maintenance will far outweigh the high initial investment.

3. TRAINING METHODS

3.1. CONTRACTOR TRAINING

3.1.1. Problems

High Training Costs. Contractor training can be expensive. The SARAH development team has received development methodology, language, and environment training through commercial sources. Acceptable quotes for 80 hours of Ada language training for 20 personnel ranged from $20,000 to $77,000. For this amount the vendor was required to provide equipment in-house for the practical sessions. To obtain training in current design methodologies, CCSO sent personnel to the vendor's site. The training cost was $1,125 per week for each student. Travel and lodging costs increases this amount considerably. CCSO found that it was far more economical to have the vendor train in-house if more than six personnel required training. However, if the training is conducted in-house, it must be segregated from the work environment, otherwise the training program could be severely jeopardized.

Specify Training Needs Precisely. Since there is a large range of Ada training currently available, organizations need to correctly specify their training requirements or the training received may not cover training needs. For example, when outlining the requirement for practical training, the specification should indicate that the training must be conducted with a validated Ada compiler. Several training organizations are currently using JANUS Ada compiler hosted on IBM-PCs for practical training. The cost of this training is generally lower than comparable training using validated compilers. However, since JANUS does not provide the advanced features of the Ada language, the practical exercises are limited to basic features. An organization considering vendor training should research the market well and provide a precise specification of their needs.

Evaluation. The lack of student and instructor evaluation is also a problem. Most training organizations will not volunteer to administer student tests and provide results. As previously described, there are definite benefits to providing some form of assessment. If student assessment is required, then this must be stated in the training requirements.

Procurement Problems. For government agencies there is a long lead time for procurement. This must be taken into account in the project schedule. CCSO found that it took seven months to procure training. No doubt this time could be shortened if the requirement received higher priority; however, there are fixed lead times associated with competitive acquisition. Managers must take this into account, otherwise the lack of training could severely affect the development schedule.

3.1.2. Selection

Training Sources. One of the best sources for providing details of currently available Ada training is the Catalog of Resources for Education in Ada and Software Engineering (CREASE). This publication is available for distribution through the Defense Technical Information Center (DTIC) and the National Technical Information Service (NTIS). The accession number for this document is AD A156 687. Further information on CREASE can be obtained by contacting the Ada Information Clearinghouse (AdaIC). In addition to CREASE, the AdaIC provides training information in their periodic newsletters.

Research. CREASE can provide information on training courses; however, managers should do additional research to determine whether a particular course will be applicable for their project. One of the best ways of achieving this is to talk to others who have recently undergone training. The National SIGAda conferences are a good place to do this type of research. Apart from being able to discuss training needs with the many experienced people who attend, many of the training organizations are available to discuss their training curriculums.

3.1.3. Benefits

Instructor Experience. Members undergoing vendor training can greatly benefit from the experience of the instructors. In addition to training, many organizations are also involved in the development of Ada systems. As such, the instructors gain practical design and development experience which can be relayed to the students. If the students are motivated towards applying the language rather than simply learning the syntax, experienced instructors can provide a wealth of information which will help speed the development process.

Wide Range of Courses. The Ada training community can now provide a wide range of courses. This allows organizations to more easily tailor a training program for their specific needs. For example, some organizations require more emphasis on concurrent real-time operations. Others need training in low-level features for embedded applications. Many training courses are emerging which support development with a particular methodology. Managers can select courses from a number of vendors so that the training reflects the needs of the development team.

Cost Effective. Vendor training can be cost effective if the number of people requiring training is small. In these situations, it would not be cost effective for an organization to attempt to establish a training program using in-house resources.

3.2. GOVERNMENT TRAINING

The use of government courses for Ada training is very cost effective but other factors also need consideration. CCSO received Ada management training from the Air Training Command (ATC). An Ada instructor was sent from Keesler AFB to conduct the training on-site at CCSO. As such, the only costs incurred were those for the instructor. For government organizations, government furnished training will provide the most cost effective method of training Ada personnel. However, cost is not the only consideration. The effectiveness of the training courses must also be addressed. Some of the factors affecting the overall impact of a training program are instructor experience, the range of courses that can be provided and the type of equipment used for practical training.

Currently, ATC does not provide a wide range of Ada courses. Manager training courses are available and courses on Ada programming will commence in the near future. ATC has experienced difficulties with their Ada Applications Programmer training course because of the lack of suitable equipment. Ada compilers place a high requirement on computer resources and so most machines will only support a small number of simultaneous compilations. In a training environment, students are required to complete many small programs to reinforce theoretical concepts and so several students will typically want to compile at one time. The use of microcomputers for Ada training will alleviate this problem to a large degree since each student will have the resources of one machine. The first microcomputer compiler is currently undergoing validation and when released should solve some underlying training problems. In the future, ATC will have a good range of Ada training courses covering aspects of software engineering, design methodologies, management, and language training. However, no commitment has been made to provide training on support environments and automated tools. This is an important aspect of Ada development and so these training needs need to be addressed.

CCSO experience with ATC training showed that the instructors lacked practical software development experience. The material was presented in a satisfactory manner but many of the underlying questions could not be answered because the instructor had never been employed in software development. If the Ada courses are to be effective, then the instructors must have some software development experience. CCSO found that many vendors furnished far more experienced instructors. If the instructor does not have practical experience and cannot answer questions on how the technology can be applied, then the material can just as easily be presented through video tapes, CAI packages, and technical publications.

3.3. IN-HOUSE TRAINING

3.3.1. Informal In-house Lectures

Informal in-house lectures can be effective if Ada experience already exists in the development team. CCSO attempted to establish an in-house informal lecture program for the SARAH project team but found that it was not cost effective. The main reasons for establishing this type of program were to:

- provide team members with a good insight into the Ada language prior to formal training.
- provide a means of Ada technology transfer between the SARAH project and other branches at CCSO.

The main reason for failure was that CCSO did not have personnel available with sufficient Ada experience to conduct this type of lecture program. Since the amount of time used for preparation adversely affected team productivity, the SARAH managers canceled the in-house lectures and placed more emphasis on self-study, use of the CAI package, and the viewing of video tapes.

Organizations should be careful in establishing an in-house lecture program if there is insufficient Ada and software engineering experience available. During an early experience with Ada, CCSO developed this type of program to train a team involved in evaluating the Ada language for use in digital communications applications. The personnel involved in developing the course had never received formal Ada training and based their instruction on traditional language technology. As such, the Ada software produced by the team resembled FORTRAN code. Since the team did not use any of the advanced features such as packages, generics, or tasks, the software was unstructured and difficult to understand. Moreover, the benefits of using the Ada language were not recognized since the team did not apply modern software engineering practices.

3.3.2. Computer Aided Instruction

A CAI package is very beneficial for teaching Ada syntax and for consolidation during and after formal Ada language training. The SARAH team used the ALSYS "Lessons on Ada" CAI package. This package provided a high level of training in Ada syntax. The extensive use of examples and problems make this package very effective. The interactive nature of the package also allows users to review specific areas and so it serves as a good reference source. IBM-PC compatible microcomputers were used to host the package. The package was used extensively prior to formal Ada language training so that the students could gain maximum benefit from the instructor's experience. Since they were

already familiar with much of the Ada syntax, they were able to concentrate on how the language could be used to develop the SARAH system. The ALSYS CAI package will be used throughout the SARAH project to allow software personnel to revise certain areas of the language.

The CAI package allows for training flexibility; however, usage must be controlled so that all personnel benefit from the package. Members of the SARAH team were able to organize training to suit their other work commitments. Enthusiasm for the package remains high; personnel spend a great deal of their own time training with the package. During work hours, a schedule was set up for CAI training. This was necessary so that team members could schedule their training times and so ensure that the entire team received training. An invitation was open for other members of CCSO to train with the package and so allow for technology transfer across the organization.

3.3.3. Video Tapes

Video tapes were found to be effective but were time consuming and did not allow for training flexibility. The SARAH team viewed several different series of tapes. The tapes varied in both quality and effectiveness. Time was allocated for viewing the tapes and attendance varied depending on the workload of the individual team members. The University of Houston video-taped a complete semester course on software engineering and the Ada language. Elements of this course were very beneficial; however quality was poor and it took some time to cover the major language features. A series of tapes named "The World of Ada" provided good background on the Ada language and were useful for manager education. The SARAH team also previewed "Ichbiah, Barnes and Firth on Ada". These tapes provide an in-depth introduction into Ada language syntax and could be used very effectively in an overall training program.

Tapes should be previewed and only the most appropriate tapes should be used for overall team training. To make effective use of video tapes, they should be used in conjunction with other training. Only those tapes which provide good support for particular topic areas should be viewed. A great deal of valuable time can be wasted by subjecting the entire development team to videos that do not necessarily support the overall training approach.

3.3.4. Self-Study

A library of Ada books and materials should be provided for self-study and research. In addition, the development area should have a "quiet area" where people can study without being disturbed. A project library was established soon after the commencement of the SARAH project. Team members set up a data base to control library inputs. The library consists of a

collection of Ada articles, catalogs, regulations and books. Today, the library is a very valuable asset to both the SARAH team and CCSO. Team members use the library to remain current with Ada technology advances and to familiarize themselves with various Ada features.

Access to an Ada compiler is necessary if self-study is to be of value. The SARAH team used the self-study method to learn the basic features of the Ada language. There is little point in learning features of the language if they cannot be reinforced by practical exercise problems. A TELESOFT compiler on a Burroughs XE-550 computer has been used extensively for this purpose. The self-study method has proven very effective as a prelude to formal training.

3.3.5. Attendance at Conferences

Attendance at conferences and seminars provided team members with training in the practical application of Ada technology. The SARAH team has been very active in national and local Ada organizations. These organizations provided a good forum for discussion and allowed members to gain a good insight into some of the problems and pitfalls encountered during software development. The benefits of first-hand experience are not always achievable through formal training alone.

In addition to benefits gained through active participation, conferences often provide free tutorials. The SARAH team has benefited significantly from the tutorials. Topics covered are generally applicable to practical Ada design and implementation. The knowledge gained can help speed software development and enhance software quality.

4. SUMMARY AND LESSONS LEARNED

4.1. SUMMARY

Software development using the Ada language development environment requires a high degree of training in order to achieve the full benefits designed into the system. Potential Ada developers must gain a practical insight into what is required to successfully develop Ada software. This includes the need to understand and apply the facilities of the Ada language, the various new design methodologies, the Ada programming support environment tools, and the high startup cost.

Those desiring to become Ada developers should be prepared to take full advantage of the language facilities. These facilities enhance the software engineering concepts for structured programming such as strong data typing, data abstraction, and procedural abstraction. Educators can prudently and reasonably include them in their course structure by properly partitioning the language facilities into coherent

training blocks. As other aspects of the software lifecycle environments become available such as configuration management, improved documentation techniques, testing, software reuse, and the Ada programming support environment, they should be included in a structured training program.

The development methodology selection is based on an analysis of individual needs. Software engineers must be cognizant of related methods and maintain an awareness of current and new efforts in order to take full advantage of improvements in lifecycle application techniques.

The Ada programming support environment provides productivity benefits and will become increasingly significant as the CAIS standards are implemented in industry. The understanding of this environment and its applicability are important for ensuring lifecycle integrity of software. Ada oriented conferences and seminars provide excellent forums to share information on current events in the Ada world and to find out what others are accomplishing.

Ada language training can be acquired from many different places. Training in the Ada language and its associated environment is available through commercial contractors, government, and through self-taught in-house programs. Risks associated with each must be carefully considered. High training costs, the possibility of inadequate or improper training, and procurement problems have been addressed. Care must be taken to ensure the training acquired is worth while and cost effective. The government provides some training opportunities but on a limited basis at this time. In-house training must be undertaken with the greatest of care. The Ada language environment is designed to support the most current and best software programming techniques in use today. These advanced techniques are beyond the capability of previous languages such as FORTRAN and COBOL. For example, in-house training may only teach the Ada syntax and semantics. Programmers will likely recreate FORTRAN or COBOL code in Ada which may not be as efficient as the original code. In-house training can be effective when using a combination of computer aided instruction, video tapes and self-study to supplement formal qualified training programs.

Software development using the Ada language and associated software engineering technologies will only be successful if full support is provided by management. Only then will the long term benefits be realized.

4.2. LESSONS LEARNED

CCSO has arrived at its present level of awareness about the Ada language environment through many hours of effort. The information contained in this paper represents what we considered important in gearing up for the SARAH project. Efforts prior to

the SARAH project within CCSO fell short of expectations because of a lack of understanding of the philosophy of modern software engineering and the software engineering facilities inherent in the Ada language. Self-training without that understanding proved to be useless. Subsequent formal training both commercial and government fell short of expectations. As a result of these experiences our training plan for SARAH is comprised of several methods, including the use of a CAI package, video tapes, selfstudy and active participation in local and national Ada communities. Through in-house training, team members were in a better position to take advantage of formal instruction. The formal training consisted of two 40 hour courses with a week separating them for time to exercise newly acquired knowledge.

5. CONCLUSIONS AND RECOMMENDATIONS

5.1. CONCLUSION

The Ada language and associated software technology can provide significant benefits in terms of maintainability, software reuse, and programmer productivity. However, managers must be aware that language syntax alone will not provide these benefits. If the Ada language is used without an emphasis on software engineering and without productivity tools, the software produced may be less maintainable and of poorer quality than that developed using older programming languages. Managers must be educated in this new technology if their project teams are to successfully develop Ada software. They must understand that the initial capital investment will be high. Development teams require education in the areas language training, software engineering, development methodologies, and support environments.

5.2. RECOMMENDATIONS

Recommendations are:

- Base Ada training on sound Software Engineering principles.
- Provide up front training for management.
- Research the proposed training organization for instructor experience and approach.
- Ensure that the training investment is sufficient to cover all training needs eg. design, environment, language, management, and software engineering training.
- Provide the development team with CAI packages to help consolidate language training.
- Provide quiet self-study areas . Time

should be allocated to allow team members to consolidate their training and to keep current with Ada technology.

- Support the Ada development team. Members will be required to make a significant personal effort if they are to become fully educated in the Ada environment.

A. BIBLIOGRAPHY

[1] DRUFFEL L.E., "The Potential Effect of Ada on Software Engineering in the 1980s", North Holland Publishing Company, 1983.

[2] CARLSON W.E., DRUFFEL L.E., FISHER D.A., WHITTAKER W.A., "Introducing Ada", Proceedings of ACM 80, pp 263-271, 28-30 October 1980.

[3] "Packages Spawn Ada's Growth", Software and Systems, April 1985, pp 93-100.

[4] STANLEY R.A., "Whither Ada?", DS&E, March 1985, pp 60-64

[5] BOOCH G., Software Engineering with Ada, Benjamin/Cummings Publishing, Menlo Park CA, 1983.

[6] JUDGE J.F., "Ada Progress Satisfies DOD", Defence Electronics, June 1985, pp 77-87

[7] "Ada as a Design Language", Ada as a PDL Working Group, IEEE Computer Society, 18 September 1985.

[8] "SARAH Operational Concept Document", US Air Force, 20 December 1985.

[9] WAGNER P., "Ada Education and Technology Transfer Activities", ACM Ada Letters, Vol II No 2.

[10] "Methodman", Ada Joint Program Office, National Technical Information Service (NTIS), accession number AD A123 710.

[12] BOOCH G., "Object Oriented Development", IEEE Transactions on Software Engineering, Vol. SE-12 No. 2, February 1986.

[13] CAMERON J.R., "An overview of JSD", IEEE Transactions on Software Engineering, Vol. SE-12 No. 2, February 1986.

[14] CHERRY G.W., "The PAMELA Designer's Handbook", Thought Tools, Reston Virginia.

[15] HAILPERN B., "Multiparadigm Languages and Environments", IEEE Software, Vol.3 No.1, January 1986.

[16] "Ada Language System Textbook", 1102-9, Softech Inc, Waltham MA, February 1984.

[17] "MIL-STD Common Ada Interface Set (CAIS)", National Technical Information Service (NTIS), accession number AD A157-589.

[18] CONN R., "Overview Of the DoD Ada Software Repository", Dr Dobbs Journal, February 1986.